Parallel Algorithms – Examples

v.0.4 - 21/3/2022

Michele Amoretti
Quantum Information Science
University of Parma

QIS.UNIPR.IT
QUANTUM INFORMATION SCIENCE @ UNIPR

Contents

Quantum Information Science
University of Parma
Parco Area delle Scienze
43124 Parma
Italy
http://www.qis.unipr.it

Preface

This is a collection of examples related to the Parallel Computing part of the High Performance Computing course (M.Sc. in Computer Engineering, University of Parma).

1 Parallel Algorithms

1.1 SUM and SUM2

We consider two parallel algorithms for adding the numbers contained in a finite array (whose size is n). Algorithm 1 uses $n/2$ processors. The $A[2i] + A[2i+1]$ operations are performed in parallel, for each SUM() invocation.

Algorithm 1 SUM(A)
Input: a finite array A filled with numbers
Output: the sum of the numbers contained in A

if $|A| = 1$ **then**
  return $A[0]$
else
  return $\mathrm{SUM}(\{A[2i] + A[2i+1] : i \in [0, .., |A|/2)\})$
end if

The total *work* of Algorithm 1 is

$$W(n) = 2W(n/2) + 1, \tag{1}$$

where 1 is the constant work of the **if**. We also know that $W(1) = O(1)$. Thus, we can expand the recursive equation of the work:

$$W(n) = 2W(n/2)+1 = 2(2W(n/4)+1)+1 = .. = nO(1)+n/2+n/4+..+1 = O(n). \tag{2}$$

The *depth* of the parallel algorithm is

$$D(n) = D(\lceil n/2 \rceil) + O(1) = O(\log n). \tag{3}$$

The running time of the algorithm corresponds to its depth. Its efficiency is

$$E_{\mathrm{SUM}}(n) = \frac{n-1}{\frac{n}{2}\log n} \simeq \frac{2}{\log n}, \tag{4}$$

because the number of sums in the sequential algorithm is $n - 1$.

Algorithm 2 uses $p < n/2$ processors.

Algorithm 2 SUM2(A)
Input: a finite array A filled with numbers
Output: the sum of the numbers contained in A

if $|A| = 1$ **then**
  return $A[0]$
else
  $\Delta \leftarrow \lceil n/p \rceil$
  $A[k\Delta] \leftarrow A[(k-1)\Delta + 1] + .. + A[k\Delta] \quad \forall k \in [1, .., p]$
  $\mathrm{SUM}(\{A[\Delta], .., A[p\Delta]\})$
end if

The first part of Algorithm 2 is slow, as it takes $\Delta = n/p$ time to complete. However, the second part is fast, as it takes $\log p$ time to complete. The overall running time is:

$$T_{\mathrm{SUM2}}(n,p) = n/p + \log p. \tag{5}$$

The efficiency is:

$$E_{\mathrm{SUM2}}(n,p) = \frac{n-1}{p\frac{n+p\log p}{p}} = \frac{n-1}{n+p\log p}. \tag{6}$$

If we choose p such that $p\log p = n$, we have the following consequences:

$$p \simeq \frac{n}{\log n}, \tag{7}$$

$$T_{\mathrm{SUM2}}(n) = O(\log n) \tag{8}$$

$$E_{\mathrm{SUM2}}(n) = \frac{n-1}{2n} \simeq 1/2. \tag{9}$$

In conclusion, SUM2 has logarithmic running time like SUM, but uses a sublinear number of processors. The resulting efficiency is much higher than the one of SUM and does not depend on n.

1.2 Maximal Independent Set

A maximal independent set (MIS) in an undirected graph is a maximal collection of vertices $\mathcal{I}$ such that no pair of vertices in $\mathcal{I}$ are adjacent.

Algorithm 3 is a randomized sequential approach for finding a MIS, given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. We assume there is a total ordering on $\mathcal{V}$ and we denote the neighborhood of a vertex v as $\mathcal{N}(v)$.

Algorithm 3 $\mathsf{MIS}_s(\mathcal{G} = (\mathcal{V},\mathcal{E}))$
Input: a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, with a total ordering on $\mathcal{V}$
Output: a maximal independent set $\mathcal{I} \subset \mathcal{V}$

1: $\mathcal{I} \leftarrow \emptyset$
2: **while** $\mathcal{V} \neq \emptyset$ **do**
3: randomly pick $v \in \mathcal{V}$
4: $\mathcal{I} \leftarrow \mathcal{I} \cup \{v\}$
5: $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v, \mathcal{N}(v)\}$
6: **end while**
7: **return** $\mathcal{I}$

The running time of Algorithm 3 is

$$T_{\mathrm{MIS}_s} = O(|\mathcal{E}|), \tag{10}$$

as all the edges have to be checked.

A better solution is Algorithm 4 (by Luby [1]), which is still randomized but parallel. Algorithm 4 requires $O(|\mathcal{E}||\mathcal{V}|^2)$ processors. Its running time is

$$T_{\mathrm{MIS}_p} = O(\log^2 |\mathcal{V}|). \tag{11}$$

Luby's algorithm sets the MIS problem is in $\mathbf{NC}_2$, i.e., the class of problems that can be solved in time $O(\log^2 n)$ using $O(n^2)$ processors, where n is the size of the input. It remains an open problem whether the MIS problem is in $\mathbf{NC}_1$.

Algorithm 4 $\mathsf{MIS}_p(\mathcal{G} = (\mathcal{V}, \mathcal{E}))$
Input: a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a total ordering on $\mathcal{V}$
Output: a maximal independent set $\mathcal{I} \subset \mathcal{V}$

$\mathcal{I} \leftarrow \emptyset$
$\mathcal{G}' = (\mathcal{V}', \mathcal{E}') \leftarrow \mathcal{G} = (\mathcal{V}, \mathcal{E})$
while $\mathcal{V}' \neq \emptyset$ **do**
    $\mathcal{X} \leftarrow \emptyset$
    for all $v \in \mathcal{V}$ (in parallel) **do**
        randomly choose to add v to $\mathcal{X}$ with probability $1/(2|\mathcal{N}(v)|)$
        (if $|\mathcal{N}(v)|$ then always add v to X)
    end for
    $\mathcal{I}' \leftarrow \mathcal{X}$
    for all $\{(v, w)\} \in \mathcal{X}^2$ (in parallel) **do**
        if $(v, w) \in \mathcal{E}'$ **then**
            if $|\mathcal{N}(v)| \leq |\mathcal{N}(w)|$ **then**
                $\mathcal{I}' \leftarrow \mathcal{I}' \setminus \{v\}$
            else
                $\mathcal{I}' \leftarrow \mathcal{I}' \setminus \{w\}$
            end if
        end if
    end for
    $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}'$
    $\mathcal{Y} \leftarrow \mathcal{I}' \cup \mathcal{N}(\mathcal{I}')$
    $\mathcal{V}' \leftarrow \mathcal{V}' \setminus \mathcal{Y}$
end while
return $\mathcal{I}$

References

[1] M. Luby, *A Simple Parallel Algorithm for the Maximal Independent Set Problem* SIAM Journal on Computing, vol. 15, no. 4, pp. 1036–1053, 1985.